Attorney Docket No.: D02316-04 PATENT

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

Inventor: Eric J. Sprunk)
)
)
U.S. Serial No.: 09/827,630)
) Art Unit: 2135
Filed: April 6, 2001)
) Examiner: Ponnoreay Pich
)

Title: AUTHORIZATION USING CIPHERTEXT TOKENS

DECLARATION UNDER 37 C.F.R. § 1.131

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir,

I, Eric J. Sprunk, hereby declare as follows:

1. I am the named and true inventor in the above referenced patent application and that I am the sole inventor of the subject matter disclosed and claimed in the above referenced patent application.

2. I submitted a description of my invention, now claimed in claims 1-7 and 11-14 of the above application, to the law department of General Instrument Corporation in an "Invention Record Form." I signed the Invention Record Form on October 5, 1999 and the signatures on the Invention Record Form are my own. A copy of the Invention Record Form is provided with this declaration as Attachment A. General Instrument Corporation Invention Record Form No. D02316CIP4.

3. I conceived the invention recited in claims 1-7 and 11-14 of the above application prior to June 2, 1998. The conception of the invention prior to this date is attested to in paragraph III(9) of the aforementioned General Instrument Corporation Invention Record Form No. D02316CIP4, and evidenced by the June 2, 1998 General Instrument Memorandum entitled "Application Security for TCI". This memorandum was referenced in and physically attached to General Instrument Corporation Invention Record Form No. D02316CIP4 when the form was witnessed by Alexander Medvinsky, a General Instrument Corporation employee, on November 5, 1999. See Attachment A.

4. I constructively reduced my invention to practice prior to June 2, 1998, and this reduction was memorialized in the aforementioned "Application Security for TCI" memorandum. This memorandum was provided to fellow General Instrument employees Paul Moroney, Gary Albeck, B. Meandija, Petr Peterka, Xin Qui, Stuart Moskovics, Steven Anderson, K. Miller, J. Fellows, Annie Chen, Lawrence Tang, Mark DePietro, Douglas Makofka, Reem Safadi, and Lawrence Vince (as evidenced by the distribution list on the face of the memorandum).

5. Upon information and belief, the date of receipt of General Instrument Corporation Invention Record Form No. D02316CIP4 by the General Instrument Corporation law department was October 8, 1999, as evidenced by the "General Instrument Corporation Intellectual Property" date stamp on the first page of Attachment A.

6. I hereby declare that all statements made herein based upon knowledge are true, and that all statements made based on upon information and belief are believed to be true,. These statements were made with the knowledge that willful false statements and

the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Dated: 27 Jan 2006

By: ______________
Eric J. Sprunk

APPENDIX A

General Instrument Corporation Invention Record Form No. D02316CIP4
Inventor: Eric J. Sprunk

U.S. Serial No.: 09/827,630

General I. .trument Corporation®
Intellectual Property Department
For Internal Use Only

Invention Record Form
GI Docket No. D2316 CIP4

18926-003160

I. **Administrative Information**

1. Short Descriptive Title of the Invention: ~~Object & Process Security~~ Authorization Using Ciphertext Tokens

2. Identify all persons who contributed to this invention, including persons from other divisions and/or outside companies:

	Inventor 1	Inventor 2
Full Legal Name	Eric Sprunk	
Home Address	6421 Cayenne Lane	
City, State, Zip	Carlsbad, CA 92009	
Citizenship	US	
Division/Co. Location	AT (SD)	
Office Phone No.	858-404-2426	
Mgr.'s Name & Phone No.	Moroney 858-404-2448	
Signature of Inventor		
Date	10/5/99	

	Inventor 3	Inventor 4
Full Legal Name		
Home Address		
City, State, Zip		
Citizenship		
Division/Co. Location		
Office Phone No.		
Mgr.'s Name & Phone No.		
Signature of Inventor		
Date		

3. □ Check box if there are additional inventors listed on separate sheets. Additional information concerning inventors, if any

CONFIDENTIAL & PROPRIETARY Rev. 02/99

U.S. Serial No.: 09/827,630

Invention Record Form

II. Background Information

1 Do you believe this invention was developed while working under or in the performance of experimental, developmental or research work called for by a government contract or with the understanding that a government contract would be awarded? ☒ No □ Yes If yes, please explain:

2. Has your invention been disclosed to anyone outside General Instrument in a speech, exhibit, presentation, product, product manual, report, lecture, trade show, technical article, publication or otherwise? □ No ☒ Yes If yes, please explain:

ATT/TCI - June 98 - UNDER NDA

3. Is this invention related to any previous GI invention disclosures of which you are aware (made by you or someone else)? □ No ☒ Yes If yes, please explain:

D2303 D2315 D2310 } Complementary inventions
D2311 D2317 D2312)

4 Name of product(s) and/or project(s) for which this invention was developed:

DCT 5000

5. Planned or actual use of invention:

Various stages - from 4Q99 through 2002

6. What economic benefits do you think GI can derive from this invention?

~~[illegible]~~ Potential that all CA manufacturers (competitors, RB, etc) may implement this CA/security model. Copy protection in general [illegible] (SDMI)

7 When do you expect a product incorporating this invention to be sold, offered for sale or shown to someone outside of GI? (If a product or prototype has already been sold, offered for sale or shown, please identify the earliest date this happened.)

4Q99 for some aspects.

8 Has a working model of the invention been built and tested (or appropriate software been written)?
□ No ☑ Yes If yes, who has witnessed a demonstration, and when?

Some aspects for 5000 settop Sept. 99

Signature of Submitter(s) 10/5/99
Date

Read and understood by (Witness Signature(s)) 10/5/99
Date

U.S. Serial No.: 09/827,630

Invention Record Form

9. List below any patents, publications, articles, texts, products, etc. which describe technology similar to your invention including reference material which may be useful in understanding the background technology of your invention. (Use a separate sheet if necessary and attach a copy of each item. Please include copies of all bibliographical information.) (Use a separate sheet if necessary)

Only JAVA type applications may be relevant.
Nothing specific known

Signature of Submitter(s)

10/5/99
Date

Read and understood by [Witness Signature(s)]

10/5/99
Date

GI CONFIDENTIAL & PROPRIETARY

U.S. Serial No.: 09/827,630

Invention Record Form

III. Description of the Invention

1 Please provide a very brief (i.e., one short sentence) summary of your invention.

System for applying CA to [illegible] of portable software objects & resources
System applies CA to objects/resources instead of channel

2. Briefly describe the field of technology to which your invention relates

Conditional access & Security

3. Briefly describe the problems, issues or needs which led to the invention

Control of individual objects & resources rather than one big group.
Control with respect to time access
Control with respect to different platforms/systems

4. How have others addressed these problems, issues or needs?

CA in general is applied to Channel - not objects or resources
[illegible] acts as control [illegible] in [illegible] for channel [illegible] flow or no flow based on authorization

5. Describe those particular features or functions of your invention which you think may be novel or technical advancements over the technology you listed in section II.9.

Attaching/Generating CA to particular objects/resources
for set time
for different platforms/systems

6. Best Mode: Describe any and all preferences you personally have regarding how to best implement, build, produce or use your invention (e.g., preferred parts, materials, techniques, etc. which you feel are best in practicing your invention). Each submitter's opinion is important here, even if there is disagreement. Please list anything you think will make the invention better in any way.

As described in [illegible]

7. Briefly describe any alternative uses, variations or modifications of your invention which you contemplate.

described in [illegible]

8. Please provide any additional information you think should be known by the attorney reviewing this form.

None

Signature of Submitter(s) 10/5/99
Date

Alexander Medvinsky
Read and understood by (Witness Signature(s)) 10/5/99
Date

GI CONFIDENTIAL & PROPRIETARY Rev 02/98

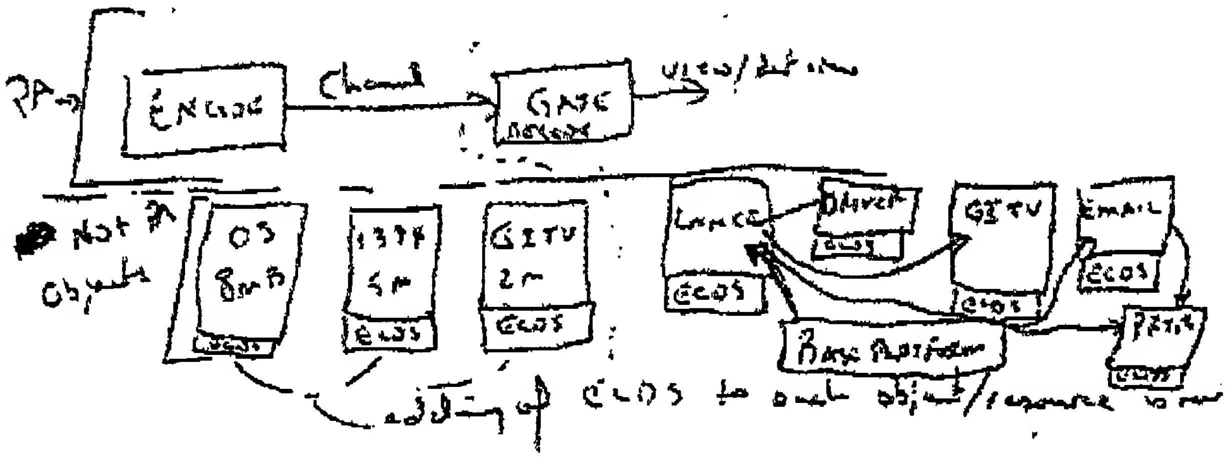
U.S. Serial No.: 09/827,630

Invention Record Form

9. Please provide a detailed description of your invention. Your description should ideally provide as many details of your invention as possible in order to achieve optimal patent protection. An ideal disclosure should describe the construction and operation of the invention including drawings (flow charts, schematics, block diagrams, mechanical drawings, photographs, etc.) and any relevant engineering laboratory notebook pages, reports, program listings, etc. If you have already prepared reports or other descriptive information, there is no need to rewrite it. Simply attach it and reference it in your invention disclosure data sheet (for example, "see attached 9 page engineering progress report addressed to John Doe dated 1 Jan., 1992 for description of amplifier circuit").

See attached 6/2/98 Memo and more specifically, security level 8 on pg. 9.

(Focus on Memos of 6/2/98 & 6/11/98 —)
From Eric Sprunk to P. Moroney et al & Sid Gregory, respectively.
both Subject line Application Security (Concepts) for TCI



adding of ECOS to each object/resource in our

Basic
Resource - anything used by an object to operate as intended — may be another object or physical device (printer).

Signature of Submitter(s)

10/5/99
Date

Read and understood by [Witness Signature(s)]

10/5/99
Date

GI CONFIDENTIAL & PROPRIETARY

Rev. 02/98

U.S. Serial No.: 09/827,630

[illegible] 6/9/98
10:51AM

Memorandum General Instrument

Date: June 2 1998

Subject: Application Security for TCI

From: Eric Sprunk

To: P Moroney, G. Albeck, B Mesandija, P. Peterka

CC: X. Qiu, S Moskovics, S Anderson, K. Miller, J Fellows, A. Chen, L. Tang, M. DePietro, D Makofka, R. Safadi, L. Vince

1. Introduction

This memo provides background to facilitate the definition of TCI requirements (and GI design decisions) associated with applications security on the DCT5000 (hereafter the "5000") product.

CONFIDENTIAL

1.1 Acronyms & Abbreviations

5000	TCI's DCT-5000 advanced settop
A&A	Authorization and Authentication, in that order
ACP	Access Control Processor
App	Application
BIOS	Built-in Operating System
CA	Conditional Access or Certificate Authority
EMM	Entitlement Management Message
ECDS	Entitlement Control Data Structure
ECM	Entitlement Control Message
ET	Execution Token
IVV	Independent Validation and Verification
JVM	Java Virtual Machine
OS	Operating System

2. Security Environments

2.1 The Video Service Security Model

A video service is a continuous stream of data consisting of individual program segments. Different security techniques apply control to this situation:

- Encryption is used through the possession of a valid key.
- The encrypted stream is routed through an Access Control Processor (ACP) security device.
- The ACP only decrypts the service if:
    - it has a valid key; and
    - ECM information passes certain data checks (or gates), e.g. possessing a specific tier.
- The encryption key used is changed regularly to facilitate this, e.g. hourly or monthly

The placement of the ACP in series with the data path is crucial, as this makes its "gatekeeper" functionality possible. Were the ACP not in series with the data stream, such as with a typical DVB smart card system, then the security control effected by the ACP would need to reach outside it to another settop component. This creates security risks avoided by merging MPEG security processing with Conditional Access.

It remains possible to inject clear data downstream of the ACP, and downstream non-ACP circuitry will accept and process such data normally. With the exception of minor security mechanisms like

Macrovision copy protection, the control of security is limited to within the ACP. Components outside the ACP cannot be depended upon to enact security functions

Such a configuration seeks to control the entire data stream as its first objective, and a given piece of data from that data stream (e.g. a program) as a second objective. The ability to do this comes from the continuous series of checks performed on each program, which each presents the opportunity to change a key or tier data associated with the program. If keys or tiers are changed for the headend ACP that encrypts the service, then a settop ACP has no choice but to do the same to decrypt.

But a specific program is really "gated" only once. If ever the ACP makes the decision to decrypt that program, then all ability to control the program is thereafter lost with this creation of clear data. If that clear program is stored (on a sufficiently large media), then it will be available forever Aside from the possible unavailability of large storage media, encryption control is binary in nature and impossible to recover once lost. Consequentially, post-ACP injection of clear video data will be successful, whether that data was recorded from earlier decryption or from some never-encrypted source of MPEG data.

This existing security model for video has limitations when applied to applications. The discussion below highlights these differences one at a time, in the context of known or implied TCI requirements.

2.2 Application Security

An application (or "App" hereafter) has characteristics in common with a video program which allow video types of security control to work acceptably for purposes that follow the video model. However, the difference between an App and a video program gives rise to new problems in need of new security solutions. There are a number of these, with only partly satisfactory solutions available for some problems

An obvious example is how Apps differ from video in the size of their data. Video data streaming at even 1Mbps for a one hour program comprises 450 MB of information, storage of which tends to be impractical at present. This storage problem presently serves as a barrier to replaying video data. But, an App is comparatively tiny at less than one megabyte, and storage is clearly feasible. The replay of old Apps or the injection of new Apps will be easier than for video data, and may therefore be a more significant problem.

This problem alone illustrates how App security techniques must be extended out beyond the ACP It is a given that the ACP cannot undertake all the functions of the entire 5000 in a single chip, though this may be possible some day for a low end settop. Until such a "one chip settop" exists, new security techniques will be needed to deal with Apps. It will not be sufficient for an ACP to serve only as a data stream gatekeeper, as this will not address the newly significant problem of data replay and control outside the ACP.

Some extensions of ACP security to outside the ACP are easy to identify, and have identifiable security benefits and limitations in addressing some requirements. For others this is not so straightforward, and careful consideration of requirements, the practicality of available security solutions, security benefits, and limitations is needed. Some App security problems are very difficult to counter without difficult and significant development efforts A discussion of value and possible diminishing return is paramount.

This memo elucidates multiple possible levels of App security on the 5000. The models are described starting at the same level as video security, then through several increasing levels of protection The models are listed sequentially, and are taken directly from or heavily implied in the TCI DCT5000 specification. In some cases, the requirement listed is derived from GI interpretation of TCI's high level security goals for the 5000 The definition of each requirement is in italics, followed by discussion in normal typeface.

3. The Levels of Application Security

The single high level requirement to "secure Applications" can readily be resolved into a number of specific sub-levels to consider one at a time In general, only the first of these levels can be attained without extending security outside the ACP Further, an Entitlement Control Data Structure[1] (ECDS) appended to each App is needed for any Authorization functions to occur, and includes a digital signature. The Operating System (OS) must make security checks at various times using this data structure, and use the result of the check as a hard decision on certain OS functions. Several possible security levels are possible, with the assumed level of trust of the OS itself affecting them all The size of the ECDS must be considered, as well as system decisions such as repetitive download of a single App or its ECDS for security purposes. It is also necessary to distinguish the initial launch of an App from its continuous execution afterwards. These possibilities are all discussed below Each security level includes the functions of levels below; e g Level 4 App Security includes all techniques and protections described for Levels 1, 2, and 3.

3.1 Level 1: Encrypted Application Download

This security level is defined as controlling the entry of an App into a settop via cable data pathways. Download is here defined as being the movement of MPEG data over the FDC or OOB, and does not include the injection of data directly into App memory by another path.

This level is the easiest to attain, since a stream of data comprising an App can be treated identically to a video stream. The App stream may be inband or out of band, but it can be simply placed on an MPEG PID in encrypted form, with associated ECMs. The single EMM stream that conveys all entitlements to individual settops is used to convey encrypted App entitlements as well.

The encrypted App passes through the ACP[2] in the normal manner, and decryption occurs only if authorized[3]. Clear, Fixed Key, and Full Encryption modes would be available.

3.2 Level 2: Download Authentication

Authentic Apps are here defined as Apps that are approved by the network operator using a digital signature.

The network operator would authenticate an App by processing it in the headend or elsewhere, and by appending such authentication to the App via the ECDS. Such network operator authentication includes the entitlements or authorizations needed to use that App. It is absolutely crucial that only the network operator be capable of authenticating an App.

[1] The Entitlement Control Data Structure is analogous to an ECM for a video service, and conveys the entitlements needed by an ACP to authorize that specific App.

[2] Note that routing the OOB MPEG data through the ACP has both ACP and settop design implications. The ACP must receive MPEG transport both from the inband and OOB sources, which requires that it have two MPEG inputs, or that MPEG data be multiplexed outside the ACP

[3] Encrypted download has a form of Authentication, where only the possessor of the encryption key can mark an App as authentic, or confirm authenticity This type of Authentication is implicit, and may be sufficient for a form of basic protection.

A digital signature is the obvious way of achieving this, with a network operator control computer holding the key that creates this signature[4] Either symmetric (e.g. DES based) or asymmetric (e g RSA or DSA or ECC based) signatures would work, but asymmetric signatures offer the best solution. The choice of asymmetric signature type must be made based on speed, signature size, licensing, and other considerations not discussed in this section

Defining an Authentication mechanism does not address the circumstances under which the Authentication is confirmed. The App can be checked for validity at different times that define different security levels. These are enumerated in sections below *Application Security Level 2 only assumes basic authentication, such as right after download decryption and before the App is loaded into storage.*

Since App signature verification is almost inescapably linked to Authorization, it must occur within the ACP. Signature verification in the ACP also minimizes the burdensome impact of hash and signature functions on the same CPU that performs video, administrative, or GUI functions. Secure confirmation and reportback of signature verification failures are important, and there is an implied requirement for Return Path capability Settops without RP or phone modem reportback capability cannot be monitored for App security behavior, and represent a higher level of risk[5]

The consideration of Authentication brings us to the most important issue in App security Current approaches[6] to Authentication have the OS itself performing verification and enforcement in the event of failure. This is because the security benefit of Authentication is inherently dependent upon the trust level of the OS. One might think that an ACP can block execution of an App by the OS, but this is untrue so long as OS design and operation is itself beyond ACP secure control. If the OS circumvents an Authentication check, then there is nothing whatsoever that the ACP can do about it. In fact, the ACP would not even be aware of such an event. The trust level of the OS will be a recurring theme in this discussion, and will be returned to again in a subsequent section.

3.3 Level 3: Authenticated Launch

Authenticated Launch is defined as the initiation of execution of an App by the OS only if it is network operator approved.

This level is where the Authentication defined in the previous section is verified by the OS, using the ACP, at App launch time. Authentication of launch requires the following steps:

1. The OS loads 100% of App data into the ACP. The signature is not initially included. The OS must not execute the App until the ACP has confirmed that the signature is valid.

2. The ACP forms the Message Authentication Code (MAC) using a hash function.

3. The OS separately loads the App signature into the ACP.

[4] At this security level, Authorization functions are not present for an Application. An App digital signature would thus only authenticate the App itself. At a higher security level described later, both the App and its authorization requirements would be included in the digital signature.

[5] Should Applications be allowed in devices such as this? Perhaps the ability to prohibit them in the event of future problems is needed, to at least allow problem settops (with a hacked OS) to be identified if necessary

[6] Microsoft Authenticode is a digital signature scheme with Microsoft taking on the role of the network operator. MS signs the applications using Authenticode, and MS OS products confirm that signature as a pre-condition for using the MS Crypto API, rather than as a pre-condition for launching a program. MS Authenticode does not perform any Authorization functions

4. The ACP checks the digital signature, responding with VERIFIED or FAILED

5. If VERIFIED, the OS initiates execution of the App.

6. If FAILED, the OS erases the App from executable memory

3.4 Level 4: Authorized Launch

Authorized Launch is defined as the *<u>initiation</u>* *of execution of an App by the OS only if appropriate encryption keys and entitlements are possessed by that specific settop ACP*

Authorization of launch can be achieved by at least two approaches with different levels of practicality. An App is always stored in authenticated form, but it could be stored either in encrypted form, or in clear form. (Note that, if stored in encrypted form, the need for encryption during download may be obviated.) If Apps are stored encrypted, then they must be decrypted to allow execution. This requires that:

1. the OS load all App data into the ACP

2. the ACP decrypt all App data.

3. the ACP hand all decrypted App data back to the OS.

The first step of this 3 step process occurs during Authentication, but the second and third steps do not and represent additional work. Authentication takes care of perhaps 50% of the work of decryption.

The security benefit of encrypted storage is dependent upon the existence of Authentication:

- Lacking Authentication, encrypted storage prevents injection of unapproved Apps into the settop, since the encryption key is not (normally) available for illicit use.

- With Authentication present, encrypted storage has value only if the OS is untrustworthy.

But this is a contradiction. If the OS is untrusted, then Authentication has no value, so encrypted storage actually adds nothing to security. This conclusion is typically illustrative of OS importance in evaluating App security. ***<u>Almost</u> all possible App security fails if the OS is not trustworthy.*** Encrypted storage is not recommended, assuming (inescapable) Authentication is present.

Assuming no encrypted storage, Authorization of launch requires the following steps:

1. The OS loads the ECDS into the ACP.

2. The ACP checks whether it possesses the entitlements and keys necessary to run that App

3. The ACP advises the OS of AUTHORIZED or NOT AUTHORIZED status.

4. The OS loads and begins execution of the App if AUTHORIZED.

5. The OS erases the App and does not execute if NOT AUTHORIZED.

Recall that this security level presumes the protections present in lower security levels, including Authentication. Both Authentication and Authorization checks must occur before launch, but in which order? In general, Authentication will require much more time to perform than Authorization, so determination of NOT AUTHORIZED status is faster if authorization is done first

The checks for Authentication and Authorization could be combined into a single LAUNCH or DO NOT LAUNCH status from the ACP, but this adds little. In the event of a DO NOT LAUNCH decision by the ACP, it is important to distinguish whether Authentication or Authorization failed. Authorization failures can mean that a current EMM is needed, while failed Authentication can mean memory corruption or an illicitly injected App.

It is recommended that Authorization, then Authentication, be checked in a combined process that should be obvious from the above. The status returned to the OS would be either NOT AUTHORIZED, AUTHORIZED BUT FAILED, or AUTHORIZED AND PASSED

3.5 Level 5[7]: OS Execution Epochs

OS Execution Epochs are here defined as the OS-initiated, timed cessation of continuously running App code <u>days or weeks</u> after launch, analogous to video Category Epochs.

Execution security is here differentiated from launch security because they are subtly different problems with different levels of threat, and because securing execution is harder than securing launch. Since it is still fundamentally the behavior of the OS *itself* that is being secured, OS trust remains central.

Launch security can be achieved by a gating function between App storage and the execution memory of the nonsecure CPU. The OS can be designed to perform Authorization and Authentication ("A&A") checks with the ACP before allowing an App through this gate into execution memory, which is defined as Level 4 Application Security. Once in execution memory, this gate is not available. This leaves us with two problems:

- It is not possible to expire long-running Apps, such as Email[8]. If a settop is activated and the Email App began legally, it is desirable for Email to stop execution if it is not paid for in the future. If no check is performed after App execution begins, then this is no longer possible. With a check, key and data changes in the ECDS would "age" a long-running App and force it to be subscribed to again.
- An App substitution attack is possible if an App can be overwritten in execution memory at a carefully-timed place in its execution. That App could then be replaced by an illicit App. This is a slightly different attack than replacing a not-yet-loaded App stored in flash or on a hard disk, which is easier. Other OS manipulations may also allow an illicit App to replace a legitimate one. We wish to periodically confirm the authenticity of running[9] Apps.

If neither of these problems are of concern, then App Security above Level 4 is not needed. If these are of concern, then a means of checking and potentially stopping a running App with the ACP is needed after the App is launched. Level 5 App Security in this memo assumes a coarse resolution mechanism to enforce the expiration of an App, as would be required to cease a monthly subscription.

Authentication and Authorization can be repeated in the background at some rate, even after an App is launched. This can be done either on some fixed regular basis, or after a period programmed at

[7] Again, this level includes the protections of security levels described above. Since both Authorization and Authentication are part of Level 4, Level 5 and higher assumes that both are required.

[8] The presumed business desirability of doing this gives this level of Application security its name

[9] This implies that a running App must still be in a form that allows it and its ECDS to be re-authenticated. The ECDS must be in memory, the sequence of App and ECDS information must not have changed, and no temporary runtime data structures can be inserted anywhere among these

time of App launch, or upon ACP demand. *Level 5 App Security is defined as fixed or programmed infrequent post-launch Authentication, with the OS being trusted to properly perform the operation* The OS could unconditionally repeat A&A on a regular basis, such as daily at night, or could observe the expiration time for a given App and delete it at that point.

At Level 5, the ACP is not designed to expect A&A or report back any failure of the OS to perform A&A. The ACP will never know if background A&A is occurring or not. But, the ACP must be the source of reference time[10] for the OS during time calculations used to determine the expiration of an App This is because a trivial attack exists, where the OS has its time reference altered to continously push the point of expiration into the future so it never arrives.

3.6 Level 6: OS Application Pay Per View

OS Application PPV (APPV) is here defined as the OS-initiated, timed cessation of continuously running App code minutes or seconds after launch, as with video Free Preview type functions.

Level 5 App security left us with a problem. It is not possible to secure the Application equivalent of Pay-Per-View with a Free Preview Period. We would want a short period of allowed use to elapse, followed by cessation of function and the presentation of a purchase decision to the user. If this is not of concern, Level 6 App Security is not needed. *If any of these are of concern*, then a means of checking a running App with the ACP is needed with fairly fine resolution in time.

An approach of regular or programmed infrequent A&A will not suffice for this, and the OS must undertake additional function that is more purely Conditional Access related. To enable APPV, the following steps are needed:

- The App ECDS must label a point in time we are familiar with, the end of Free Preview
- The OS must track the passage of time to detect the end of Free Preview
- At the end of Free Preview, the OS must:
    - suspend App execution.
    - initiate a purchase GUI screen with the ACP.
- If the user buys, a purchase debit is made with the ACP.

An unlikely alternative exists for Application PPV. Often software is written in two forms, one being a "teaser" demo form and the other a product release. If this same (storage consuming) approach is used, then the demo software would be equivalent to a Free Preview. If the product release is only launched after an IPPV purchase, then video IPPV has been imitated. Note that the demo software could be run by the user indefinitely, however, so the imitation is imperfect.

[10] Secure time from the ACP must come from secured messages distributed within the network, and from ACP internal time tracking functions. Secure Time Messages must be digitally signed and verified by the ACP, and the ACP must always advance time forward and never backward. The ACP can also maintain a time reference of perhaps dubious accuracy which is used to coarsely adjust ACP internal time in lieu of a better reference such as system time

3.7 Level 7: ACP Watchdog & Reportback

ACP Watchdog & Reportback is here defined as the ACP tracking[11] OS security activities where possible, and securely reporting the results of such monitoring back to a network operator controller The ACP directs the OS to submit any of these Apps to the ACP for A&A on demand.

Level 6 App security left us with a problem. It is not possible for the ACP to tell if the OS is enforcing the expiration of Applications, whether they are short term APPV or long term subscription oriented expirations.

If this is not of concern, Level 7 App Security is not needed. If this is a concern for any OS function, then a means for the ACP to monitor and report back the behavior of the OS is needed. The ACP must be able to shadow this process and potentially detect OS failure to properly perform. In the case of short term or long term expiration of Apps, this could take a form similar to current IPPV processing, where the ACP tracks elapsed time. If the OS does not run A&A for a given App by some maximum time, a failure is detected and reported back by the ACP.

However, since the ACP has no App decryption function at that time[12], it can do no more than record that no purchase apparently occurred. The ACP is unable to enforce security against the OS. The trust level of the OS again determines security success or failure.

3.8 Level 8: ACP Execution Token

It is desirable to involve the ACP in the actual mandatory process of executing an application. To do this, we have mentioned the encryption of an entire App as one method in previous sections, but that is limited and impractical due to the processing required to decrypt large amounts of data. For APPV, it is further desirable for a portion of an App to be capable of running up to the point where user purchase is mandatory. (Encrypted code cannot run.) This is similar to the demo software model mentioned in Level 6 above.

An Execution Token (ET) is one solution to these problems. The ET is no more than a code segment that is encrypted in some simple manner[13], with the decryption information held by the ACP until a purchase of that App is made by the User through the OS. ACP and OS design are complicated by this, as Execution Tokens would need to be stored, processed, and passed around by both. In particular, ACP memory would swell, probably to an uncomfortable level.

3.9 Level 9: ACP Memory Guardian

This level of Application security is a fantasy at current levels of technology and return on a security development investment. It is included here to establish an upper bound on Application security

[11] This definition is only viable if the total number of Apps tracked by the ACP is reasonable, such as 16 or perhaps 32. If beyond this number, then internal memory requirements for an ACP security asic grow too large. Though effort can be made to minimize App record sizes, there will be a practical upper limit, so system security design must handle this constraint

[12] The Application has already been loaded into execution memory and launched, in order to begin the "Free Application Preview Period" in the first place.

[13] A simple keystream generator is sufficient, with the keystream handed out to the OS for XOR onto the segment of the App. DFAST is a candidate that allows GI patent enforcement if needed. A programmable relative address within the App, plus a segment length, plus an App ID, plus a key is needed within the ACP

The ACP can conceptually be extended to completely manage the entire downloadable memory space, so that no App exists, is loaded, is launched, or is allowed to continue running without the ACP having the unstoppable opportunity to erase it.

4. Related Considerations & Comments

4.1 Trust Levels for an OS

There are three obvious security worries for an OS. First, the OS could be manipulated in some way to evade a security function, or be tricked into making a different decision than normal for a security check. A historical example of this might be a security check implemented in a batch file under DOS DOS allows the press of keys Control C to abort execution of a batch file, so it is possible to evade any check made in a batch file by carefully timed key presses.

The second problem is unsatisfactory design of the OS itself. This could occur if:

1. A pirate alters OS code to do something different that its designers put into it; or

2. The OS designers made a mistake during design; or

3. The OS designers intentionally designed around inconvenient functions, or placed "back doors" into the code for business or surreptitious employee reasons.

The third problem is intentional misdesign of the OS, either by a malicious employee of the OS vendor, or by conscious predatory intent of the OS vendor company.

These three cases[14] each mandate increasingly extreme and laborious measures to counter them:

1. The OS must be digitally signed, and this signature checked using the BIOS and ACP

2. The OS must be thoroughly reviewed at the source code level, and tested at the executable code level. This should preferably occur prior to release, by a disinterested party highly skilled in software design and Independent Validation and Verification (IVV) processes.

3. The thorough review of (2) above must include monitoring the creation of releasable code The actual process of compiling source code into executable code must be witnessed and very closely monitored by an independent party. The exact source code used in compilation must be stored outside the control of the OS vendor, then compilation observed, then executable code digitally signed and similarly stored. Under no circumstances can the OS vendor ever possess the digital signing key.

4. Various OS double checking roles for the ACP have been mentioned in this memo. The more the OS is of concern, the more weight should be given to implementing these

4.2 Securing Functions Outside the Application Layer

So far we have focused on Applications, which are the highest level software objects running in the 5000 They often interface with the user, perform GUI functions, etc. But other software objects in

[14] Note that some OS vendors have been known to have very buggy software, to have frequently misdesigned software with unintended function, and to be predatory against other companies

the 5000 are also the subject of security Items like a TCPIP stack, telephone or cable modem driver, or other lower level software can be secured for sale through A&A.

This is achievable through the same mechanisms as securing Apps. The CA system and its ACP have no awareness of OS or software details, only that the A&A process moves certain specific data around for processing in specific ways. It makes no difference to A&A if the segment of data being Authorized and Authenticated comprises any of the following:

- Applications
- Protocol stacks
- Hardware drivers
- Data files
- A Java Virtual Machine
- A Java applet

All can be secured by the same CA system mechanisms and cryptography. Data is data, and the nature of the data is transparent to the CA system. But impact outside the CA system is huge.

Each and every software object to be subjected to A&A must have an ECDS. The functions of A&A described in this memo must be added to BIOS and OS, and to any software that manipulates other software. For example, it must not be possible to launch the equivalent of a small OS as an App, as that App-OS-on-top-of-an-App can itself launch Apps that are not subject to A&A. (This is Java!)

4.3 A Warning Regarding the Java Virtual Machine

The Java Virtual Machine (JVM) is an Application level artifice that serves as an OS of sorts all by itself. Java Applets run on the JVM, and these Apps are as significant to the network operator's business as are non-Java Apps. Java Applets must be secured using the exact same techniques, and to the exact same security level as regular Apps, without exception.

For every paragraph in this memo that has the words "Operating System" or "OS", a mirror paragraph stating Java Virtual Machine can be written. Securing the Java Applets is just as paramount, and if they are ignored, then there is little point in securing regular Applications.

This is true even if a JVM is not present in the original 5000 product launch. When a JVM is added at a later time (perhaps as a downloaded APPV App), then it must be a JVM designed to work with the ACP at the security level selected for the entire 5000 system.

5. Summary

Though securing Applications and other software objects is easy to conceive for a CA system, this is absolutely not the case for the rest of the system!! In fact, the design impact of securing any software object whatsoever as described in this memo is substantial. The precepts and paradigms of current software engineering practice view the security techniques discussed herein as complete and horrible anathema. It must not be underestimated how mind-bending a change this represents. Radical new approaches will naturally be met with resistance, and will encounter unforeseen problems, and will require substantial development effort and time. Consideration of